Multi-task Inference of Diffusion Networks

Anonymous Author(s)

ABSTRACT

To infer structures in diffusion networks based on observed diffusion results, existing approaches customarily infer each diffusion network in isolation, and assume that the observation data for each inference task is sufficient. In many real-world situations, it is common to observe diffusion results on multiple diffusion networks with similar structures, while the amount of observation data collected on each network is often limited. In this work, we study how to infer multiple similar diffusion networks jointly with limited observation data for each network. To this end, we introduce an iterative strategy which in turn updates the inference results for all diffusion networks by exploiting the similarity between the networks, and theoretically guarantee the monotonicity and convergence of the iterative process. Extensive experimental results on both synthetic and real-world networks verify the effectiveness and efficiency of our approach.

CCS CONCEPTS

• **Computing methodologies** → **Maximum likelihood modeling**; • **Mathematics of computing** → **Computing most probable explanation**.

KEYWORDS

Diffusion Network, Influence Relationship, Multi-task Inference

ACM Reference Format:

Anonymous Author(s). 2024. Multi-task Inference of Diffusion Networks. In *Proceedings of 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24).* ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

A diffusion network is a directed graph, where a directed edge from a parent to a child indicates that the parent node can influence the child node with a certain probability [3, 18]. The structure of a diffusion network delineates the underlying influence relationships between network nodes. Such a structure is essential for understanding the intrinsic diffusion mechanisms and developing strategies to control future diffusions on the network [5]. Nonetheless, diffusion network structures are often not naturally accessible, and need to be inferred based on diffusion results observed from history [15]. The objective of the diffusion network inference task is to learn the diffusion network structure (i.e., the parent-child influence relationships) from the observation data. This problem has received considerable attention in areas such as information propagation [13], viral marketing [23], and epidemic prevention [32].

To infer diffusion network structures, most existing approaches resort to precise timestamps of historical node infections (known as cascades) [3, 4, 6–10, 21, 24–27, 29, 30], since they assume that nodes infected sequentially within a short time interval are more likely to possess influence relationships. Nevertheless, in reality, tracing node infection timestamps as cascades is not always feasible or affordable, especially in diffusion processes with long-term durations and wide node distributions, such as the spread of epidemics. Furthermore, due to some unavoidable objective factors such as a long incubation period (i.e., the time from infection to illness), the observed cascades are unlikely to pinpoint the exact occurrence time of infections [11]. Given the limitation of cascade-based approaches, a few approaches try to infer diffusion network structures without cascades, using only the eventual infection statuses of nodes in historical diffusion processes [1, 11, 18, 19], since observing the final infection statuses of nodes at the end of each diffusion process is more feasible than constantly monitoring the nodes to trace exact infection timestamps.

Both types of approaches infer diffusion network independently, and assume that sufficient observation data for each inference task is available, from which a reliable structure can be inferred. In many real-world situations, however, it is difficult to gather enough observation data on every diffusion network. Fortunately, observation data can be gathered on multiple closely related diffusion networks. For instance, at different periods, the influence relationships among certain individuals often have a lot in common but not be identical, forming a set of diffusion networks with similar structures, and at each period, the amount of available observation data on each network is often limited. For this kind of situations, multi-task learning [34] suggests that it may be possible to achieve more accurate inference results by leveraging the relatedness across multiple related inference tasks. Nevertheless, so far, few studies have effectively leveraged the similarity across multiple diffusion networks to infer more accurate diffusion network structures. Although several existing work can infer multiple diffusion networks simultaneously [16, 33], they still deal with each inference task independently, and can not take advantage of the similarity across diffusion networks.

In this work, we investigate the problem of how to infer multiple similar diffusion networks jointly with limited observation data for each network. We propose an effective and efficient algorithm called MIND (which is an anagram of the bold letters in **M**ulti-task **I**nference of **D**iffusion **N**etworks) for this problem. Equipped with a probabilistic generative model, MIND is able to estimate the likelihood of all inferred network structures jointly, by exploiting the similarity between the networks. In addition, MIND adopts an iterative strategy which in turn updates each inferred network structure in each iteration. It can be theoretically guaranteed that the likelihood of inferred network structures increases monotonically through the iterations of MIND, and the iteration process terminates in a finite steps.

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

The remainder of the paper is organized as follows. We first review the related work, and present our problem statement. Then, we introduce our proposed MIND algorithm, followed by reporting experimental results and our findings before concluding the paper.

2 RELATED WORK

Existing approaches to diffusion network inference can be briefly classified into two groups, namely, (1) cascade-based approaches, and (2) infection status-based approaches.

2.1 Cascade-based Approaches

Most existing approaches to diffusion network inference assume that the exact timestamp of each node infection in historical diffusion processes is available. With these node infection timestamps as cascades, they infer influence relationships between nodes by identifying frequent sequences in cascades. According to the type of their inference methods, these cascade-based approaches can be categorized into two main types, namely, (1) optimization approaches, and (2) embedding approaches.

The optimization approaches often transform the problem of diffusion network inference into an optimization problem, by modeling the likelihood of cascades with a convex optimization objective function. Then, they find the optimal diffusion network structure, which is most likely to generate cascades, with diverse techniques, such as sequential quadratic programming [6, 25], the EM algorithm [33], stochastic gradient methods [3, 9], block coordinate descent [4], survival theory [8], sparse recovery [29], neural mean-field dynamics [12], or decoupling into multiple parallelizable subproblems [21, 26]. Besides using convex optimization objective functions, several optimization approaches adopt an objective function with the property of submodularity. Then, they approximate the optimal solution by performing greedy algorithms. NetInf [7] and MulTree [10] are two state-of-the-art approaches of this type. The main difference between them is that NetInf considers only the most probable propagation tree to achieve high efficiency, while MulTree considers all propagation trees supported by each cascade to achieve high accuracy.

Inspired by the idea of metric learning [17], embedding approaches try to project nodes of the objective diffusion network into a new embedding space. The strength of the influence relationship is measured by calculating the Euclidean distance between the nodes in the embedding space. These methods commonly employ uniform distributions [2] or kernel functions [14] to model the strength of the influence relationship between nodes, and learn model parameters based on cascades. Although embedding methods cannot explicitly reveal the structure of the diffusion network, they allow the user to visualize the influence relationships between nodes by visualizing a low-dimensional embedding space.

Both of the above two types of approaches demand complete cascades. It has been proven that with sufficient complete and correct cascades, diffusion network structures can be inferred accurately by using some simple inference methods [21]. In addition, several methods have been proposed to mitigate the effects of partially incorrect [20, 31] or missing cascades [5, 15].

Departing from these cascade-based approaches that customarily focus on infer a single diffusion network, our MIND algorithm is designed for inferring multiple diffusion networks with similar structures, and is based on only the final infection statuses of nodes.

2.2 Infection Status-based Approaches

As tracing cascades is often infeasible due to high cost, a few studies investigate how to infer diffusion networks based on final infection statuses of nodes in historical diffusion processes, which are more accessible in practice. According to the type of inference methods, they can be grouped into two main types, namely, (1) lifting effect-based approaches, and (2) bayesian inference-based approaches.

The lifting effect-based approaches [1] calculate the lifting effect of each seed node u to another infected node v based on the set of initially infected nodes and the set of eventually infected nodes. They keep on adding a directed edge (i.e., an influence relationship) from u to v if u has the current greatest lifting effect to v, until the number of edges reaches a user-specified value. Although lifting effect-based approaches tend to be very efficient, they require to know the initially infected nodes in each diffusion process, which may be untraceable in some real-world scenarios.

The bayesian inference-based approaches [5, 11, 16, 18, 19] model the target diffusion network as a bayesian network. They investigate the joint influence effects of the parent nodes set under different combinations of infection statuses on the infection state of a child node based on historical data of node infection statuses, and reconstructed the diffusion network based on bayesian inference techniques. Due the solid mathematical bases, a few bayesian inference-based approaches, such as TENDS [11], provide a theoretical guarantee on their accuracy performance. Nevertheless, the number of all possible combinations of infection status data grows exponentially with the number of parent nodes, so that this kind of approaches become complicated and computationally expensive when the target diffusion network has a dense structure.

Most of infection status-based approaches aim to infer a single diffusion network. Only the LMDN algorithm [16] tries to infer a multi-aspect diffusion networks, in which each aspect refers to a single diffusion network. LMDN assumes that different aspects have different structures, and infers each diffusion network separately. In contrast, our MIND algorithm assumes that the target multiple diffusion networks share similar structures, and leverages the structure similarities to improve inference accuracy.

3 PROBLEM STATEMENT

A diffusion network can be represented as a directed graph. In the graph, a directed edge from a parent node to a child node represents an influence relationship between them, which indicates that when the parent node is infected and the child node is uninfected, the parent will successfully infect the child with a certain probability. Formally, suppose there are k diffusion networks, namely, $G_1, ..., G_k$, directed graph $G_t = (V, E_t)$ ($t \in \{1, ..., k\}$) denotes the t-th diffusion network, where $V = \{v_1, v_2, ..., v_n\}$ is the set of n nodes, and E_t is the set of directed edges in G_t. Let $F_{t,i}$ be the set of parent nodes of node $v_i \in V$ w.r.t. E_t, as each node has two possible infection statuses (i.e., infected status, which is denoted by 1, and uninfected status, which is denoted by 0), there are $2^{|F_{t,i}|}$ possible combinations of the infection statuses of v_i's parent nodes, where $|F_{t,i}|$ refers to the number of nodes in set $F_{t,i}$. Let X_i and $X_{F_{t,i}}$ be the infection

status variable of node v_i and the infection status variables of nodes in $F_{t,i}$, respectively, and $\pi_{t,i,j}$ be the j-th possible combinations of the infection statuses of nodes in $F_{t,i}$ ($j \in \{1, ..., 2^{|F_{t,i}|}\}$), we denote $\theta_{t,i,j,1}$ as the probability of v_i being infected (i.e., $X_i = 1$) under the condition that the infection statuses of nodes in $F_{t,i}$ are instantiated to the j-th possible combination (i.e., $X_{F_{t,i}} = \pi_{t,i,j}$).

In this paper, we focus on inferring multiple similar diffusion networks jointly with limited observation data for each network. In this problem, we assume that the k objective diffusion networks $G_1, ..., G_k$ have similar structures, and the observation data contains only the eventual infection statuses of nodes in historical diffusion processes. Then, our problem statement can be formulated as follows.

Given: k sets $D_1, ..., D_k$ of infection status results observed on k diffusion networks $G_1, ..., G_k$, where $D_t = \{S_t^1, ..., S_t^{\beta_t}\}$ is the infection status results observed on $G_t = (V, E_t)$ in β_t diffusion processes ($t \in \{1, ..., k\}$), and $S_t^\ell = \{s_{t,1}^\ell, ..., s_{t,n}^\ell\}$ is a n-dimensional vector that records the eventual infection status $s_{t,i}^\ell \in \{0, 1\}$ of each node $v_i \in V$ in the ℓ-th diffusion process ($\ell \in \{1, ..., \beta_t\}$).

Infer: the edge set E_t of each diffusion network G_t.

4 THE MIND ALGORITHM

In this section, we first explain how to measure the likelihood of inferred diffusion network structures jointly, followed by introducing how to update each inferred network structure in an iterative way. Then we present the detailed steps of the MIND algorithm, and conclude this section with a complexity analysis on the algorithm.

4.1 Measuring Likelihood of Inferred Structures

Given the infection status results $D_1, ..., D_k$, the multi-task inference of diffusion networks is equivalent to the problem of finding k diffusion networks $G_1, ..., G_k$ that maximize the following posterior probability:

$$G_1, ..., G_k = \arg \max_{G_1,...,G_k} p(G_1, ..., G_k \mid D_1, ..., D_k). \tag{1}$$

According to Bayes rule, we have

$$\begin{aligned} &p(G_1, ..., G_k \mid D_1, ..., D_k) \\ &= \frac{p(G_1, ..., G_k)p(D_1, ..., D_k \mid G_1, ..., G_k)}{p(D_1, ..., D_k)}. \end{aligned} \tag{2}$$

Since each set of infection status results D_t ($t \in \{1, ..., k\}$) takes place on corresponding network G_t independently, we can reformulate $p(D_1, ..., D_k \mid G_1, ..., G_k)$ as follows.

$$p(D_1, ..., D_k \mid G_1, ..., G_k) = \prod_{t=1}^{k} p(D_t \mid G_t). \tag{3}$$

As there is no prior knowledge on the probability distribution of $D_1, ..., D_k$, we equally treat each possible value of $p(D_1, ..., D_k)$ by assuming that $p(D_1, ..., D_k)$ is equal to a certain constant. Then, the problem in Eq. (1) is equivalent to the following problem.

$$G_1, ..., G_k = \arg \max_{G_1,...,G_k} \mathcal{L}(G_1, ..., G_k) \tag{4}$$

where $\mathcal{L}(G_1, ..., G_k) = p(G_1, ..., G_k) \prod_{t=1}^{k} p(D_t \mid G_t)$.

As it is difficult and expensive to find the globally optimal $G_1, ..., G_k$, we adopt a local search method. To be specific, during the search of the structure of G_t ($t \in \{1, ..., k\}$), we fix the structures of the other networks with their current inference results. With this method, we infer the structure of each network in turn. In this situation, $\mathcal{L}(G_1, ..., G_k)$ can be reformulated as follows.

$$\mathcal{L}(G_1, ..., G_k) = p(D_t \mid G_t)p(G_t \mid G_{-t})Q(G_{-t}), \tag{5}$$

where $G_{-t} = \{G_1, ..., G_{t-1}, G_{t+1}, ..., G_k\}$ refers to the remaining networks after removing G_t from $\{G_1, ..., G_k\}$, and $Q(G_{-t}) = p(G_{-t}) \prod_{s \neq t} p(D_s \mid G_s)$.

The three parts in $\mathcal{L}(G_1, ..., G_k)$ work differently:

(1) Probability $p(D_t \mid G_t)$ measures the likelihood that the infection status result D_t is generated by G_t. According to Eq. (3) in TENDS [11], $p(D_t \mid G_t)$ can be calculated as:

$$p(D_t \mid G_t) = \prod_{i=1}^{n} \prod_{j=1}^{2^{|F_{t,i}|}} \prod_{r=0}^{1} \theta_{t,i,j,r}^{N_{t,i,j,r}}. \tag{6}$$

where $\theta_{t,i,j,1}$ is the probability of v_i being infected under the condition that the infection statuses of nodes in $F_{t,i}$ are instantiated to the $\pi_{t,i,j}$, and $N_{t,i,j,r}$ is the number of times situations $X_i = r \wedge X_{F_{t,i}} = \pi_{t,i,j}$ appear in D_t.

(2) Probability $p(G_t \mid G_{-t})$ encodes how similar network G_t and the other networks should be. Due to the similarity across the networks, if an edge frequently appears in the other networks, the probability of this edge existing in G_t is high. Let $e_{t,i,j} \in \{0, 1\}$ indicates whether there is an influence relationship from v_i to v_j in G_t (1 for yes and 0 for no), the probability that directed edge (v_i, v_j) exists in G_t can be estimated as follows.

$$p(e_{t,i,j} = 1 \mid G_{-t}) = \frac{\sum_{s \neq t} e_{s,i,j}}{k - 1}, \tag{7}$$

A slight limitation manifests itself in the fact that the estimated probability could be 0 or 1, when an edge exists in none or all of the other networks. In these two situations, the estimated probability for this edge will be fixed at 0 or 1 without any change, and thus fails to help update the inference result of the corresponding edge in G_t. To help the probability estimated by Eq. (7) fall in (0, 1), we introduce a perturbation factor α in the numerator and a perturbation factor $\tau\alpha$ in the denominator. Subsequently, the estimated probability can be reformulated as follows.

$$p(e_{t,i,j} = 1 \mid G_{-t}) = \frac{\sum_{s \neq t} e_{s,i,j} + \alpha}{k - 1 + \tau\alpha}, \tag{8}$$

where α is a small value (α is set to 0.0001 in this paper), and $\tau > 1$ is a hyperparameter (τ is set to 2 in this paper). These perturbation factors can effectively prevent $p(e_{t,i,j} \mid G_{-t})$ from being 0 or 1, thus facilitating the exploration of similarity across the networks. As $p(e_{t,i,j} = 0 \mid G_{-t}) = 1 - p(e_{t,i,j} = 1 \mid G_{-t})$, probability $p(G_t \mid G_{-t})$ can be calculated as follows.

$$\begin{aligned} &p(G_t \mid G_{-t}) \\ &= \prod_{i=1}^{n} \prod_{v_j \in F_{t,i}} p_{t,j,i} \prod_{v_j \notin F_{t,i} \cup \{v_i\}} (1 - p_{t,j,i}), \end{aligned} \tag{9}$$

where $p_{t,j,i} = p(e_{t,j,i} = 1 \mid G_{-t})$.

(3) $Q(G_{-t})$ is independent w.r.t. G_t.

We can iteratively perform the local search method to approximate the optimal solution. Before the first iteration, we initialize the inference result for the structure of each network G_t ($t \in$

$\{1, ..., k\}$) by executing an existing infection status-based approach TENDS [11] with infection status result D_t. Let $G_1^{(T)}, ..., G_k^{(T)}$ be the inferred networks after T iterations. In $(T+1)$-th iteration, when the first $t-1$ networks have been updated as $G_1^{(T+1)}, ..., G_{t-1}^{(T+1)}$, what we need to do is finding a $G_t^{(T+1)}$ that maximizes the value of the following objective function.

$$G_t^{(T+1)} = \arg\max_{G_t} f_t^{(T+1)}(G_t), \tag{10}$$

where $f_t^{(T+1)}(G_t)$ represents the formula below:

$$p(D_t \mid G_t) p(G_t \mid G_1^{(T+1)}, ..., G_{t-1}^{(T+1)}, G_{t+1}^{(T)}, ..., G_k^{(T)}).$$

We would like to point out that the $G_t^{(T+1)}$ found by Eq. (10) tends to increase the likelihood of inferred diffusion network structures. This nice property can be explained by the following Lemma 1. Note that all the proofs related to this section are given in the appendix.

Lemma 1. *If $G_t^{(T+1)}$ is the solution obtained by solving the optimization problem in Eq. (10), then*

$$\begin{aligned} &\mathcal{L}(G_1^{(T+1)}, ..., G_{t-1}^{(T+1)}, G_t^{(T)}, G_{t+1}^{(T)}, ..., G_k^{(T)}) \\ \leq &\mathcal{L}(G_1^{(T+1)}, ..., G_{t-1}^{(T+1)}, G_t^{(T+1)}, G_{t+1}^{(T)}, ..., G_k^{(T)}). \end{aligned} \tag{11}$$

Combining Eq. (6) and Eq. (9) into $f_t^{(T+1)}(G_t)$, we have

$$\begin{aligned} &\ln f_t^{(T+1)}(G_t) \\ =&\sum_{i=1}^{n} \sum_{j=1}^{2^{|F_{r,i}|}} \sum_{r=0}^{1} N_{t,i,j,r} \ln \theta_{t,i,j,r} + \\ &\sum_{i=1}^{n} \sum_{v_j \in F_{t,i}} \ln \frac{\sum_{s \neq t} \hat{e}_{s,j,i} + \alpha}{k-1+2\alpha} + \\ &\sum_{i=1}^{n} \sum_{v_j \notin F_{t,i} \cup \{v_i\}} \ln \frac{k-1-\sum_{s \neq t} \hat{e}_{s,j,i} + \alpha}{k-1+2\alpha} \end{aligned} \tag{12}$$

where $\hat{e}_{s,j,i} = e_{s,j,i}^{(T+1)}$ for $s \in \{1, ..., t-1\}$, and $\hat{e}_{s,j,i} = e_{s,j,i}^{(T)}$ for $s \in \{t+1, ..., k\}$. A greater value of $\ln f_t^{(T+1)}(G_t)$ indicates a higher likelihood of the inferred structure of G_t. Here, we obtain the final optimization objective as Eq. (12), which could help us find the optimal $G_t^{(T+1)}$ in the $(T+1)$-th iteration.

4.2 Updating Inferred Structures

To maximize the value of $\ln f_t^{(T+1)}(G_t)$, we should find for each node v_i an optimal parent node set $F_{t,i} \subseteq V \setminus \{v_i\}$ that maximizes the following scoring function.

$$\begin{aligned} &g_t^{(T+1)}(v_i, F_{t,i}) \\ =&\sum_{j=1}^{2^{|F_{t,i}|}} \sum_{r=0}^{1} N_{t,i,j,r} \ln \frac{N_{t,i,j,r}}{N_{t,i,j}} + \\ &\sum_{v_j \in F_{t,i}} \ln \frac{\sum_{s \neq t} \hat{e}_{s,j,i} + \alpha}{k-1+2\alpha} + \\ &\sum_{v_j \notin F_{t,i} \cup \{v_i\}} \ln \frac{k-1-\sum_{s \neq t} \hat{e}_{s,j,i} + \alpha}{k-1+2\alpha}. \end{aligned} \tag{13}$$

Algorithm 1 Updating algorithm for G_t

Input: D_t, G_1, ..., G_k.
Output: G_t.

1: **for** $v_i \in V$ **do**
2: $g_m = g_t^{(T+1)}(v_i, F_{t,i})$, $F'_{t,i} = F_{t,i}$;
3: **for** $v_j \in V \setminus \{v_i\}$ **do**
4: **if** $v_j \in F_{t,i}$ **then**
5: $F_{new} = F_{t,i} \setminus \{v_j\}$;
6: **else**
7: $F_{new} = F_{t,i} \cup \{v_j\}$;
8: **end if**
9: **if** $g_t^{(T+1)}(v_i, F_{new}) > g_m$ **then**
10: $g_m = g_t^{(T+1)}(v_i, F_{new})$, $F'_{t,i} = F_{new}$;
11: **end if**
12: **end for**
13: **end for**
14: $E_t = \{(v_j, v_i) \mid v_i \in V, v_j \in F'_{t,i}\}$;
15: **return** $G_t = (V, E_t)$.

Then, the structure inference task for G_t is decomposed into finding each node a set of optimal parent nodes:

$$\max_{F_{t,i}} g_t^{(T+1)}(v_i, F_{t,i}). \tag{14}$$

This is a discrete optimizing problem. As there are 2^{n-1} possible solutions for this problem, it is too expensive to enumerate all of them. Therefore, we adjust the optimization problem in Eq. (14) as a constrained optimization problem:

$$\begin{aligned} \max_{F_{t,i}} \quad & g_t^{(T+1)}(v_i, F_{t,i}) \\ \text{s.t.} \quad & |F_{t,i} \setminus F_{t,i}^{(T)}| + |F_{t,i}^{(T)} \setminus F_{t,i}| \leqslant 1, \end{aligned} \tag{15}$$

where $|A \setminus B|$ represents the number of elements in A but not in B. This constraint specifies that in the $(T+1)$-th iteration of local search method, the $F_{t,i}$ should be updated by removing at most one node from, or adding at most one new node into, v_i's parent node set $F_{t,i}^{(T)}$ in T-th iteration.

Based on the scoring function in Eq. (13) and parent node set updating method specified by the constraint in Eq. (15), we propose the updating algorithm for G_t in Algorithm 1.

Algorithm 1 takes as inputs the infection status result D_t observed on G_t, and current inferred networks $G_1, ..., G_k$. For each node $v_i \in V$, it keeps searching a new parent node set with a higher score by removing one node from (line 5), or adding one new node into (line 7), current parent node set (lines 2–13). Then, the algorithm updates the directed edge set E_t of G_t based on the updated parent node sets (line 14), and finally, returns the updated network G_t (line 15).

In the $(T+1)$-th iteration of local search method, the $G_t^{(T+1)}$ updated by Algorithm 1 tends to increase the likelihood of inferred diffusion network structures. This nice property can be explained by the following Lemma 2.

Algorithm 2 The MIND algorithm

Input: $D_1, ..., D_k$.
Output: $G_1, ..., G_k$.

1: Use TENDS to initialize G_t with D_t, $\forall t \in \{1, ..., k\}$;
2: $change = true$;
3: **while** $change = true$ **do**
4: $change = false$;
5: **for** $t \in \{1, ..., k\}$ **do**
6: Using Algorithm 1 to update G_t as G'_t;
7: **if** $G'_t \neq G_t$ **then**
8: $change = true$;
9: $G_t = G'_t$;
10: **end if**
11: **end for**
12: **end while**
13: **return** $G_1, ..., G_k$.

Lemma 2. *Assume that the input of Algorithm 1 are* D_t, $G_1^{(T+1)}$, ..., $G_{t-1}^{(T+1)}$, $G_t^{(T)}$, $G_{t+1}^{(T)}$, ..., $G_k^{(T)}$ *and the output is* $G_t^{(T+1)}$, *then*

$$\begin{aligned} &\mathcal{L}(G_1^{(T+1)}, ..., G_{t-1}^{(T+1)}, G_t^{(T)}, G_{t+1}^{(T)}, ..., G_k^{(T)}) \\ \leqslant &\mathcal{L}(G_1^{(T+1)}, ..., G_{t-1}^{(T+1)}, G_t^{(T+1)}, G_{t+1}^{(T)}, ..., G_k^{(T)}), \end{aligned} \tag{16}$$

where the equal sign holds iff $G_t^{(T+1)} = G_t^{(T)}$.

4.3 Algorithm

Based on the updating algorithm for G_t, we propose the MIND algorithm to infer multiple similar diffusion networks jointly with limited observation data for each network.

The MIND algorithm, outlined in Algorithm 2, takes as inputs the infection status results $D_1, ..., D_k$. It first uses TENDS to initialize the structure of each network $G_t (t \in \{1, ..., k\})$ (line 1), then updates networks $G_1, ..., G_k$ by Algorithm 1 in turn and checks whether there are changes in the updated results (lines 3–12). If updated results no longer change, they are returned by MIND as the final inferred networks $G_1, ..., G_k$ (line 13).

The MIND algorithm adopts the local search method, and infers $G_1, ..., G_k$ in an iterative way. The following Theorem 1 shows that with the increasing of iteration times, the likelihood of network structures inferred by MIND monotonically increases.

Theorem 1. *Assume that* $G_1^{(T)}, ..., G_k^{(T)}$ *are the networks inferred by MIND in the* T*-th iteration, and* $G_1^{(T+1)}, ..., G_k^{(T+1)}$ *are the networks inferred by MIND in the* $(T+1)$*-th iteration, then*

$$\mathcal{L}(G_1^{(T)}, ..., G_k^{(T)}) \leqslant \mathcal{L}(G_1^{(T+1)}, ..., G_k^{(T+1)}), \tag{17}$$

where the equal sign holds iff $G_t^{(T+1)} = G_t^{(T)}, \forall 1 \leqslant t \leqslant k$.

Furthermore, the following Theorem 2 theoretically guarantee the convergence of MIND algorithm.

Theorem 2. *The MIND algorithm terminates in a finite steps.*

4.4 Complexity Analysis

In the updating algorithm for G_t (i.e., Algorithm 1), the most computationally expensive process is calculating the scoring function $g_t^{(T+1)}(v_i, F_{t,i})$ for each v_i and a possible parent node set $F_{t,i}$. The time complexity of calculating $g_t^{(T+1)}(v_i, F_{t,i})$ is about $O(\beta_t 2^\eta + k\eta)$, where β_t is the number of historical diffusion processes in D_t, k is the number of networks, and η is the upper bound of the parent node set size. Since η is determined by TENDS, the value is about $\log \beta_t$ and is much less than the number n of network nodes. Due to the constraint in Eq. (15), the possible parent node sets for each v_i can be enumerated, and the amount is about $n - 1$. Therefore, the time complexity of updating algorithm for G_t is about $O(\beta_t 2^\eta n^2 + k\eta n^2)$.

In MIND algorithm, using TENDS to initialize each G_t ($t \in \{1, ..., k\}$) takes $O(\beta_t n^2 + \eta^2 m^\eta n \beta_t)$ time, $m \ll n$ is the number of candidate parent nodes for each node pruned by TENDS. In the while loop, the most computational expensive process is using Algorithm 1 to obtain $G'_1, ..., G'_k$, which takes $O(k\beta 2^\eta n^2 + k^2 \eta n^2)$ time, where $\beta = \max\{\beta_1, ..., \beta_k\}$. Therefore, the overall complexity of MIND is about $O(k\beta n^2 + k\eta^2 m^\eta n\beta + k\beta 2^\eta n^2 T + k^2 \eta n^2 T)$, where T is the number of iterations in MIND.

5 EXPERIMENTAL EVALUATION

In this section, we first introduce the experimental setup, and then verify the effectiveness and efficiency of our MIND algorithm on synthetic and real-world networks. We investigate the effects of the number of networks, similarity between networks, network size, initial infection ratio, the amount of diffusion processes, and the iterations of MIND, on the accuracy and runtime performance of MIND. All algorithms in the experiments are implemented in Python, running on a desktop PC with Intel Core i9-9900k CPU at 3.60GHz and 32GB RAM. The source code of the MIND algorithm and data sets used in the experiments are available at https://github.com/DiffusionNetwork/MIND.

5.1 Experimental Setup

Network. We adopt LFR benchmark graphs [22] as the synthetic networks. By setting different generation parameters, such as the number n of nodes and the average degree of each node, we generate a series of LFR benchmark graphs with properties summarized in Table 1. In order to generate multiple networks with similar structures for multi-task inference, we utilize LFR benchmark graphs as original networks, and then generate similar networks by randomly removing a few directed edges from original networks. Table 2 summarizes the properties of the generated similar networks. The similarity between two networks G and G' is measured by vertex/edge overlap [28], which can be calculated as

$$similarity(G, G') = 2 \times \frac{|V \cap V'| + |E \cap E'|}{|V| + |V'| + |E| + |E'|}. \tag{18}$$

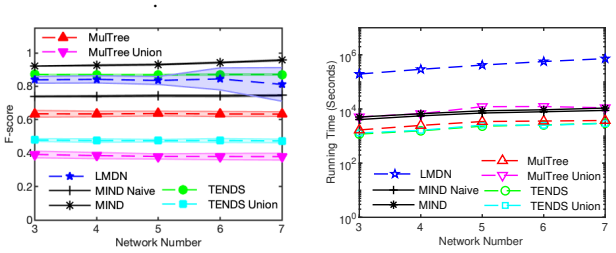
In addition, we also generate two groups of similar networks with the following two real-world microblogging networks [33], i.e., (1) DUNF, which contains 750 users and 2974 following relationships, and (2) DPU, which contains 1038 users and 11385 following relationships. For each real-world network, we randomly remove 1/3 of the edges to create five similar networks for testing. These two groups of similar diffusion networks are referred to as 5-DUNF

Table 1: Properties of LFR benchmark graphs

Graphs	Number n of Nodes	Average Degree
LFR1-5	1000, 1500, 2000, 2500, 3000	4

Table 2: Properties of synthetic networks

Similar Networks	Original Network	Similarity
G1-5, G6-10, G11-17, G18-22, G23-27	LFR3	0.6, 0.65, 0.7, 0.75, 0.8
G28-32, G33-37, G38-42, G43-47	LFR1, LFR2, LFR4, LFR5	0.7



(a) Accuracy **(b) Running time**

Figure 1: Effect of the number of networks

and 5-DPU, respectively. Similar methods for generating multiple objective diffusion networks are commonly used in existing related studies [16, 33].

Infection Status Data. The infection status results $D_1, ..., D_k$ can be generated by simulating β times of diffusion processes on each network with randomly selected initially infected nodes in each simulation. In each diffusion process, each infected node tries to infect its uninfected child nodes with a certain probability, which subjects to a Gaussian distribution with a mean of 0.3 and a standard deviation of 0.05, so that about 95% of values of the probability are within a range from 0.2 to 0.4 [5, 11, 15, 16]. Besides infection status results, the corresponding cascades (i.e., infection timestamps) are also recorded for cascade-based algorithms in the experiments. Similar generation methods for infection status data and cascades are commonly used in existing studies [1, 5, 7, 11, 15, 16, 18, 19, 33].

Performance Criterion. We evaluate the performance of MIND in terms of the accuracy of inferred structure. For each objective network, the accuracy of structure inference can be measured by the F-score (i.e., the harmonic mean of precision and recall) of the inferred directed edges, which can be calculated as follows.

$$F\text{-}score = \frac{2 \cdot precision \cdot recall}{precision + recall},$$

$$precision = \frac{N_{TP}}{N_{TP} + N_{FP}}, \ recall = \frac{N_{TP}}{N_{TP} + N_{FN}},$$

where N_{TP} is the number of true positives, i.e., the true edges that are correctly inferred by the algorithm, N_{FP} is the number of false positives, i.e., the wrong inferred edges which are not in the real network, and N_{FN} is the number of false negatives, i.e., the true edges which are not correctly inferred by the algorithm. We report the average F-score on all objective diffusion networks as the accuracy performance of structure inference. A greater value of F-score indicates a more accurate result of structure inference.

Benchmark Algorithms. We compare MIND with MulTree [10], which is a classical cascade-based approach, and TENDS [11], which is a high-performance infection status-based approach. Since MulTree requires users to specify the number of edges to be inferred, we give it the privilege to use the actual number of edges in the network as input. As MulTree and TENDS infer each objective network independently, we also run MulTree and TENDS with an union strategy (referred to as MulTree Union and TENDS Union), in which infection data collected from all objective networks are concatenated into a single data set, and only one network is inferred based on this single data set. In addition, we compare MIND with LMDN [16], which is a state-of-the-art algorithm for multi-aspect diffusion network inference. For LMDN, we regard each objective network as one aspect of a multi-aspect diffusion network. Furthermore, to verify the effectiveness of the MIND algorithm in terms of exploiting the similarity across the networks, we execute a naive version of MIND (denoted as MIND Naive), which does not utilize any knowledge from other inferred networks by setting $\hat{e}_{s,j,i} = 0$, $\forall s, i, j$, in Eq. (13).

5.2 Effect of Number of Networks

To study the effect of the number of networks on algorithms' performance, we test MIND and benchmark algorithms on five groups of synthetic networks, i.e., G11-13, G11-14, G11-15, G11-16, G11-17, where the number of networks varies from 3 to 7. We simulate 300 diffusion processes on each network (i.e., β = 300). In each simulation, $0.15n$ nodes are randomly selected as initially infected nodes.

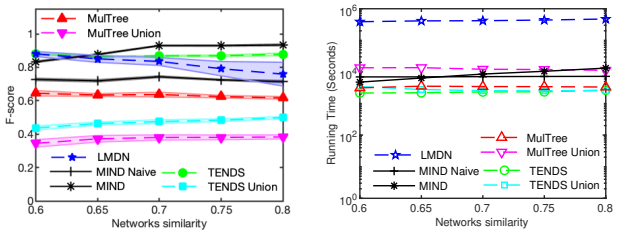
Fig. 1 reports the accuracy performance (standard deviation is showed by shaded area) and running time of each algorithm, from which we have the following observations.

(1) MIND has the highest accuracy, and tends to achieve slightly higher accuracy when there are more networks for multi-task inference. This is because that MIND can effectively leverage the similarity across networks, and more networks brings more references for MIND to conduct multi-task inference.

(2) The increasing number of networks has little effect on the accuracy of MulTree, MulTree Union, TENDS, TENDS Union, and MIND Naive. The reason behind is that these algorithms carry out inference tasks in isolation, and thus can not take advantage of the similarity across networks. Furthermore, the low accuracy of MulTree Union and TENDS Union indicates that the union strategy is not a good idea for inferring multiple similar networks.

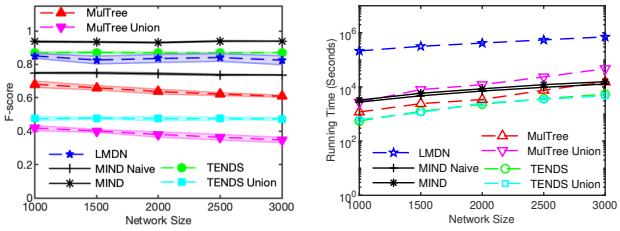
(3) More networks degrade the robustness of LMDN. Because for LMDN, more networks increase the risk of estimating wrong aspect labels for infection status results. When LMDN uses wrong infection status results to infer network structures, performance degradation happens.

(4) MIND is much faster than LMDN, and shows competitive efficiency performance, compared with other tested algorithms. The running time of each algorithm increases with the number of networks.



(a) Accuracy (b) Running time

Figure 2: Effect of diffusion network similarity



(a) Accuracy (b) Running time

Figure 3: Effect of diffusion network size

5.3 Effect of Similarity between Networks

To study the effect of the similarity between networks on algorithms' performance, we test MIND and benchmark algorithms on five groups of synthetic networks, i.e., G1-5, G6-10, G11-15, G18-22, G23-27, with similarities varying from 0.6 to 0.8. We simulate 300 diffusion processes on each network (i.e., β = 300). In each simulation, $0.15n$ nodes are randomly selected as the initially infected nodes.

Fig. 2 illustrates the performance comparison results, from which we can have the following observations.

(1) MIND generally achieves the best accuracy, with TENDS and LMDN slightly better in one setting, i.e., similarity between networks being 0.6. This is because that MIND leverages the similarity between networks to improve its inference results. A small similarity will restrict the above effect. In contrast, when the similarity becomes greater, this restriction will be removed, and MIND tends to achieve higher accuracy.

(2) The accuracy of TENDS, MulTree, and MIND Naive is insensitive to the similarity between networks. Because the three algorithms infer networks independently.

(3) A greater similarity between networks improves the accuracy of MulTree Union and TENDS Union. This is because that MulTree Union and TENDS Union infer only one network. If the objective networks are similar to each other, the difference between the objective networks and the inferred network tends to be reduced.

(4) A greater similarity between networks degrades the accuracy and robustness of LMDN. Because for LMDN, networks with more similar structures increase the risk of estimating wrong aspect labels for infection status results, causing more wrong inference results.

(5) MIND is much faster than LMDN, and shows competitive efficiency performance, compared with other tested algorithms. An increasing similarity between networks results in more running time for MIND and LMDN, and has little effect on the running time of the other tested algorithms.

5.4 Effect of Network Size

To study the effect of diffusion network size on algorithms' performance, we test MIND and benchmark algorithms on five groups of synthetic networks, i.e., G28-G32, G33-G37, G11-G15, G38-G42, G43-G47, with network sizes varying from 1000 to 3000. We simulate 300 diffusion processes on each network (i.e., β = 300). In each simulation, $0.15n$ nodes are randomly selected as the initially infected nodes.

Fig. 3 illustrates the performance comparison results, from which we can obverse that MIND outperforms the benchmark algorithms in terms of accuracy, and we can also have the following observations.

(1) A larger network size tends to degrade the accuracy of MulTree and MulTree Union, and has little effect on the accuracy of the other tested algorithms. The reason behind is that MulTree considers all propagation paths, if the objective network contains more nodes, there will be significantly more possible propagation paths, which will increase the difficulty of the inference work of MulTree.

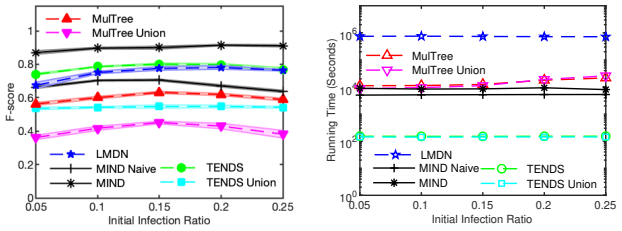
(2) The running time of each algorithm increases with the growth of diffusion network size, and MIND has better running time performance than LMDN.

5.5 Effect of Initial Infection Ratio

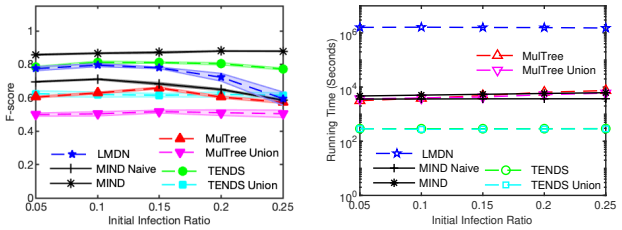
The ratio of initially infected nodes may affect the number of final infected nodes in a diffusion process. To study the effect of initial infection ratio on algorithms' performance, we test MIND and benchmark algorithms on two groups of networks, i.e., 5-DUNF and 5-DPU. We vary the initial infection ratio from 0.05 to 0.25 with an interval of 0.05. We simulate 150 diffusion processes on each network (i.e., β = 150).

Fig. 4 illustrates the performance comparison results, from which we can observe that MIND performs better than the benchmark algorithms in terms of accuracy, and we can also have the following observations.

(1) The accuracy of MIND is reasonably insensitive to the initial infection ratio. In contrast, the initial infection ratio shows significant effect on the benchmark algorithms. This is because the final infected nodes caused by a small fraction of initially infected nodes fail to comprehensively span all nodes within a target network. Consequently, the available infection data proves insufficient for benchmark algorithms to reveal all influence relationships among the nodes. An appropriate initial infection ratio ensures that the eventual spread of infection adequately traverses the majority of the objective network, supplying ample infection data to help benchmark algorithms approximate its best accuracy performance. Nonetheless, a large initial infection ratio can easily lead to false statistical correlations of node infections, which may reduce the accuracy of the benchmark algorithms. While MIND

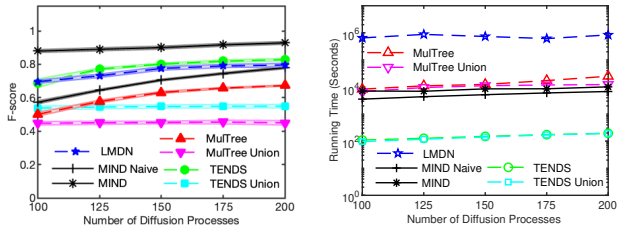


(a) Accuracy (5-DUNF) (b) Running time (5-DUNF)

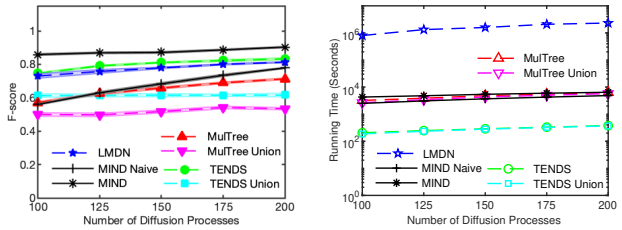


(c) Accuracy (5-DPU) (d) Running time (5-DPU)

Figure 4: Effect of initial infection ratio



(a) Accuracy (5-DUNF) (b) Running time (5-DUNF)



(c) Accuracy (5-DPU) (d) Running time (5-DPU)

Figure 5: Effect of the amount of diffusion processes

can relax this effect by leveraging the infection data across multiple similar networks.

(2) An increase in the initial infection ratio may result in longer running time for MulTree and MulTree Union, and has little effect on other algorithms.

5.6 Effect of Diffusion Process Amount

The inference of diffusion networks is based on the observed diffusion results of diffusion processes. Hence, the amount of diffusion processes may affect the accuracy performance of diffusion network inference. To study the effect of diffusion process amount on algorithms' performance, we test the algorithms on two groups of networks, i.e., 5-DUNF and 5-DPU, with different amounts of diffusion processes. To this end, for each group of networks, we simulate different number β of diffusion processes (β varies from 100 to 200) on its five networks. In each simulation, $0.15n$ nodes are randomly selected as the initially infected nodes.

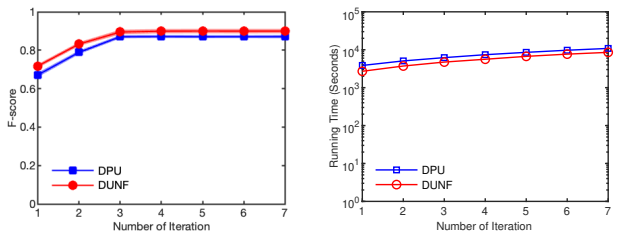
Fig. 5 illustrates the performance comparison results, from which we can observe that MIND has significantly higher accuracy even with a small amount of observation data (i.e., a small amount of diffusion processes), and we can also have the following observations.

(1) A greater amount of diffusion processes often helps the algorithms achieve more accurate results. The reason behind this is that data from more diffusion processes helps bring more information about objective networks, and helps the algorithms to have more accurate inference results.

(2) To deal with the data collected from more diffusion processes, each algorithm often requires more running time.

5.7 Effect of Iterations of MIND

To study the effect of MIND's iterations on its performance, we test MIND on 5-DUNF and 5-DPU. For each network, we simulate 150 diffusion processes (i.e., $\beta = 150$), with an initial infection ratio of 0.15 in each simulation. We vary the maximum number of iterations from 1 to 8, and report the accuracy as well as the accumulated running time of MIND at the end of each iteration.



(a) Accuracy (b) Running time

Figure 6: Effect of iterations

Fig. 6 illustrates the tested results, from which we can observe that (1) the accuracy of MIND can be improved with more iterations, and shows a reasonably fast convergence property; (2) more iterations result in more running time for MIND as a rule.

6 CONCLUSION

In this paper, we have investigated the problem of how to infer multiple similar diffusion networks jointly with limited observation data for each network. Towards this, we have proposed an effective and efficient algorithm, MIND, to solve this problem. MIND updates inferred network structures iteratively and jointly by exploiting the similarity between the networks. Experimental results on both synthetic and real-world networks have verified the effectiveness and efficiency of MIND.

REFERENCES

[1] K. Amin, H. Heidari, and M. Kearns. 2014. Learning from Contagion(Without Timestamps). In *Proceedings of the 31st International Conference on Machine Learning (ICML 2014)*. 1845–1853.

[2] S. Bourigault, S. Lamprier, and P. Gallinari. 2016. Representation Learning for Information Diffusion through Social Networks: An Embedded Cascade Model. In *WSDM 2016*. 573–582.

[3] H. Daneshmand, M. Gomez-Rodriguez, L. Song, and B. Schölkopf. 2014. Estimating Diffusion Network Structures: Recovery Conditions, Sample Complexity & Soft-thresholding Algorithm. In *Proceedings of the 31st International Conference on Machine Learning (ICML 2014)*. 793–801.

[4] N. Du, L. Song, A. Smola, and M. Yuan. 2012. Learning Networks of Heterogeneous Influence. In *Advances in Neural Information Processing Systems 25 (NIPS 2012)*. 2780–2788.

[5] T. Gan, K. Han, H. Huang, S. Ying, Y. Gao, and Z. Li. 2021. Diffusion Network Inference from Partial Observations. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI 2021)*. 7493–7500.

[6] M. Gomez-Rodriguez, D. Balduzzi, and B. Schölkopf. 2011. Uncovering the Temporal Dynamics of Diffusion Networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML 2011)*. 561–568.

[7] M. Gomez-Rodriguez, J. Leskovec, and A. Krause. 2010. Inferring Networks of Diffusion and Influence. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2010)*. 1019–1028.

[8] M. Gomez-Rodriguez, J. Leskovec, and B. Schölkopf. 2013. Modeling Information Propagation with Survival Theory. In *Proceedings of the 30th International Conference on Machine Learning (ICML 2013)*. 666–674.

[9] M. Gomez-Rodriguez, J. Leskovec, and B. Schölkopf. 2013. Structure and Dynamics of Information Pathways in Online Media. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining (WSDM 2013)*. 23–32.

[10] M. Gomez-Rodriguez and B. Schölkopf. 2012. Submodular Inference of Diffusion Networks from Multiple Trees. In *Proceedings of the 29th International Conference on Machine Learning (ICML 2012)*. 489–496.

[11] K. Han, Y. Tian, Y. Zhang, L. Han, H. Huang, and Y. Gao. 2020. Statistical Estimation of Diffusion Network Topologies. In *Proceedings of the 36th IEEE International Conference on Data Engineering (ICDE 2020)*. 625–636.

[12] S. He, H. Zha, and X. Ye. 2020. Network diffusions via neural mean-field dynamics. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*. 2171–2183.

[13] X. He, T. Rekatsinas, J. Foulds, L. Getoor, and Y. Liu. 2015. HawkesTopic: A Joint Model for Network Inference and Topic Modeling from Text-Based Cascades. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*. 871–880.

[14] S. Hu, B. Cautis, Z. Chen, L. Chan, Y. Geng, and X. He. 2019. Model-free inference of diffusion networks using RKHS embeddings. *Data Mining and Knowledge Discovery* 33, 2 (2019), 499–525.

[15] H. Huang, K. Han, B. Xu, and T. Gan. 2022. Reconstructing diffusion networks from incomplete data. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI 2022)*. 3085–3091.

[16] H. Huang, K. Han, B. Xu, and T. Gan. 2023. Multi-aspect Diffusion Network Inference. In *Proceedings of the 32nd ACM Web Conference (WWW 2023)*. 82–90.

[17] H. Huang, Y. Peng, T. Gan, W. Tu, R. Zhou, and S. Wu. 2021. Metric Learning via Penalized Optimization. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2021)*. 656–664.

[18] H. Huang, Q. Yan, L. Chen, Y. Gao, and C. S. Jensen. 2021. Statistical Inference of Diffusion Networks. *IEEE Transactions on Knowledge and Data Engineering* 33, 2 (2021), 742–753.

[19] H. Huang, Q. Yan, T. Gan, D. Niu, W. Lu, and Y. Gao. 2019. Learning Diffusions without Timestamps. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI 2019)*. 582–589.

[20] H. Huang, Q. Yan, K. Han, T. Gan, J. Jiang, Q. Xu, and C. Yang. 2023. Learning Diffusions under Uncertainty. *arXiv preprint* (2023), arXiv:2312.07942.

[21] D. Kalimeris, Y. Singer, K. Subbian, and U. Weinsberg. 2018. Learning Diffusion using Hyperparameters. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*. 2420–2428.

[22] A. Lancichinetti, S. Fortunato, and F. Radicchi. 2008. Benchmark Graphs for Testing Community Detection Algorithms. *Physical Review E* 78, 4 (2008).

[23] J. Leskovec, L. A. Adamic, and B. A. Huberman. 2007. The Dynamics of Viral Marketing. *ACM Transactions on the Web* 1, 1 (2007), 5.

[24] H. Li, C. Xia, T. Wang, S. Wen, C. Chen, and X. Yang. 2021. Capturing Dynamics of Information Diffusion in SNS: A Survey of Methodology and Techniques. *ACM Computing Surveys (CSUR)* 55, 1 (2021), 1–51.

[25] S. Myers and J. Leskovec. 2010. On the Convexity of Latent Social Network Inference. In *Advances in Neural Information Processing Systems 23 (NIPS 2010)*. 1741–1749.

[26] H. Narasimhan, D. C. Parkes, and Y. Singer. 2015. Learnability of Influence in Networks. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*. 3186–3194.

[27] P. Netrapalli and S. Sanghavi. 2012. Learning the Graph of Epidemic Cascades. In *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS 2012)*. 211–222.

[28] P. Papadimitriou, A. Dasdan, and H. Garcia-Molina. 2010. Web Graph Similarity for Anomaly Detection. *Journal of Internet Services and Applications* 1, 1 (2010), 19–30.

[29] J. Pouget-Abadie and T. Horel. 2015. Inferring Graphs from Cascades: A Sparse Recovery Framework. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*. 977–986.

[30] Y. Rong, Q. Zhu, and H. Cheng. 2016. A Model-Free Approach to Infer the Diffusion Network from Event Cascade. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM 2016)*. 1653–1662.

[31] E. Sefer and C. Kingsford. 2015. Convex Risk Minimization to Infer Networks from Probabilistic Diffusion Data at Multiple Scales. In *Proceedings of the 31st IEEE International Conference on Data Engineering (ICDE 2015)*. 663–674.

[32] J. Wallinga and P. Teunis. 2004. Different Epidemic Curves for Severe Acute Respiratory Syndrome Reveal Similar Impacts of Control Measures. *American Journal of Epidemiology* 160, 6 (2004), 509–516.

[33] S. Wang, X. Hu, P. Yu, and Z. Li. 2014. MMRate: Inferring Multi-aspect Diffusion Networks with Multi-pattern Cascades. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2014)*. 1246–1255.

[34] Y. Zhang and Q. Yang. 2021. A Survey on Multi-Task Learning. *IEEE Transactions on Knowledge and Data Engineering* 34, 12 (2021), 5586–5609.

A APPENDICES

A.1 Proof of Lemma 1

Let

$$\hat{Q} = \prod_{s=1}^{t-1} p(D_t \mid G_t^{(T+1)}) \prod_{s=t+1}^{k} p(D_t \mid G_t^{(T)}), \tag{19}$$

then

$$\begin{aligned} &\mathcal{L}(G_1^{(T+1)}, ..., G_{t-1}^{(T+1)}, G_t^{(T)}, G_{t+1}^{(T)}, ..., G_k^{(T)}) \\ =&f_t^{(T+1)}(G_t^{(T)})\hat{Q}, \end{aligned} \tag{20}$$

$$\begin{aligned} &\mathcal{L}(G_1^{(T+1)}, ..., G_{t-1}^{(T+1)}, G_t^{(T+1)}, G_{t+1}^{(T)}, ..., G_k^{(T)}) \\ =&f_t^{(T+1)}(G_t^{(T+1)})\hat{Q}. \end{aligned} \tag{21}$$

Since $G_t^{(T+1)} = \arg\max_{G_t} f_t^{(T+1)}(G_t)$, then

$$f_t^{(T+1)}(G_t^{(T)}) \leq f_t^{(T+1)}(G_t^{(T+1)}). \tag{22}$$

Thus, we have

$$f_t^{(T+1)}(G_t^{(T)})\hat{Q} \leq f_t^{(T+1)}(G_t^{(T+1)})\hat{Q}. \tag{23}$$

Combining with the equations Eq. (20) and Eq. (21) we know this lemma holds. ■

A.2 Proof of Lemma 2

Referring to the proof of Lemma 1, here we only need to prove that

$$f_t^{(T+1)}(G_t^{(T)}) \leqslant f_t^{(T+1)}(G_t^{(T+1)}), \tag{24}$$

and the equation holds iff $G_t^{(T+1)} = G_t^{(T)}$.

From Eqs. (12) and (13), we have

$$\begin{aligned} \ln f_t^{(T+1)}(G_t^{(T)}) &= \sum_{i=1}^{n} g_t^{(T+1)}(v_i, F_{t,i}^{(T)}), \\ \ln f_t^{(T+1)}(G_t^{(T+1)}) &= \sum_{i=1}^{n} g_t^{(T+1)}(v_i, F_{t,i}^{(T+1)}). \end{aligned} \tag{25}$$

Obviously, Algorithm 1 is a method for solving the optimization problem defined in Eq. (15). The candidate solution $F'_{t,i}$ is initialized as $F_{t,i}^{(T)}$ (line 2). After that, all the feasible solutions of Eq. (15) are taken into account. The candidate solution $F'_{t,i}$ would be updated only when the value of scoring function is strictly larger than that of current candidate solution $F'_{t,i}$ (lines 3-11). Given this, we have

$$g_t^{(T+1)}(v_i, F_{t,i}^{(T)}) \leqslant g_t^{(T+1)}(v_i, F_{t,i}^{(T+1)}), \tag{26}$$

the equality holds iff $F_{t,i}^{(T+1)}) = F_{t,i}^{(T)}$. Summing the above inequalities over $i = 1, ..., n$, we have

$$\sum_{i=1}^{n} g_t^{(T+1)}(v_i, F_{t,i}^{(T)}) \leqslant \sum_{i=1}^{n} g_t^{(T+1)}(v_i, F_{t,i}^{(T+1)}), \tag{27}$$

the equality holds iff $\forall 1 \leqslant i \leqslant n. F_{t,i}^{(T+1)} = F_{t,i}^{(T)}$, namely $G_t^{(T+1)} = G_t^{(T)}$. Combining with Eq. (25), we know Eq. (24) holds, and then this lemma holds. ■

A.3 Proof of Theorem 1

For t from 1 to k, using Lemma 2 we have

$$\begin{aligned} &\mathcal{L}(G_1^{(T)}, ..., G_k^{(T)}) \\ \leqslant&\mathcal{L}(G_1^{(T+1)}, G_2^{(T)}, ..., G_k^{(T)}) \\ &\dots \\ \leqslant&\mathcal{L}(G_1^{(T+1)}, ..., G_{t-1}^{(T+1)}, G_t^{(T)}, G_{t+1}^{(T)}, ..., G_k^{(T)}) \\ \leqslant&\mathcal{L}(G_1^{(T+1)}, ..., G_{t-1}^{(T+1)}, G_t^{(T+1)}, G_{t+1}^{(T)}, ..., G_k^{(T)}) \\ &\dots \\ \leqslant&\mathcal{L}(G_1^{(T+1)}, ..., G_k^{(T+1)}). \end{aligned} \tag{28}$$

There are k inequalities in (28), the conditions for the equality sign are $G_1^{(T+1)} = G_1^{(T)}$, ..., $G_k^{(T+1)} = G_k^{(T)}$, respectively. Then, $\mathcal{L}(G_1^{(T)}, ..., G_k^{(T)}) \leqslant \mathcal{L}(G_1^{(T+1)}, ..., G_k^{(T+1)})$, and the equal sign holds if and only if $\forall 1 \leqslant t \leqslant k, G_t^{(T+1)} = G_t^{(T)}$. ■

A.4 Proof of Theorem 2

The while loop updates $G_1, ..., G_k$ until the loop condition *change* = *true* not holds, which means for all $t \in \{1, ..., k\}$ that $G'_t = G_t$, i.e., through a while loop $G_1, ..., G_k$ not change. Let $G_1^{(0)}, ..., G_k^{(0)}$ be the initial value of $G_1, ..., G_k$ as line 1 in Algorithm 2. Let $G_1^{(T)}, ..., G_k^{(T)}$ be the value of $G_1, ..., G_k$ after T while loop. Then the while loop terminates when $\forall 1 \leqslant t \leqslant k, G_t^{(T+1)} = G_t^{(T)}$ holds. From Theorem 1 we know that

$$\begin{aligned} \mathcal{L}(G_1^{(0)}, ..., G_k^{(0)}) &\leqslant \mathcal{L}(G_1^{(1)}, ..., G_k^{(1)}) \\ &\leqslant \mathcal{L}G_1^{(2)}, ..., G_k^{(2)}) \leqslant \cdots \end{aligned}$$

Assume the MIND algorithm does not terminate. Then for $T = 0, 1, 2, ...$, the formula $\forall 1 \leqslant t \leqslant k, G_t^{(T+1)} = G_t^{(T)}$ does not hold, we have

$$\mathcal{L}(G_1^{(T)}, ..., G_k^{(T)}) < \mathcal{L}(G_1^{(T+1)}, ..., G_k^{(T+1)}). \tag{29}$$

Thus, we obtain a strictly increasing infinite sequence

$$\begin{aligned} \mathcal{L}(G_1^{(0)}, ..., G_k^{(0)}) &< \mathcal{L}(G_1^{(1)}, ..., G_k^{(1)}) \\ &< \mathcal{L}(G_1^{(2)}, ..., G_k^{(2)}) < \cdots \end{aligned} \tag{30}$$

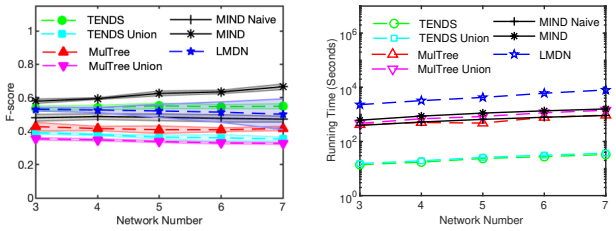
Since the possible values of each G_t is discrete and finite (with at most 2^n possible combinations), the values of $G_1, ..., G_k$ are also discrete and finite, and we can have a conclusion that the possible value of $\mathcal{L}(G_1, ..., G_k)$ is finite. If Eq. (30) is infinite, then $\mathcal{L}(G_1, ..., G_k)$ has an infinite number of values, which is a contradiction with the earlier conclusion. Therefore, The MIND algorithm terminates in finite steps. ■

A.5 Performance on ER Random Network

To verify the performance of MIND algorithm on diverse networks. We further study the effect of the number of networks on ER random networks, where the number of networks varies from 3 to 7 and the similarity is set 0.7. The number of nodes in each network is 2000, and the average degree is 4. We simulate 300 diffusion processes on each network (i.e., β = 300). In each simulation, $0.15n$ nodes are randomly selected as the initially infected nodes. The performance

comparison results is illustrated in Fig. 7, from which we can have observations that are similar to what we find in Fig. 1.

.



(a) Accuracy **(b) Running time**

Figure 7: Performance on ER random network